

Teorie automatů a formálních jazyků II

podle přednášek doc. Mojžíra Křetínského

Sazbu v L^AT_EXu připravil Dušan Dobeš

Obsah

1	Deterministická syntaktická analýza shora dolů	1
1.1	LL(k) gramatiky	1
1.2	Syntaktická analýza LL(1) gramatik	3
1.3	Analýza SLL(k) gramatik	4
1.4	Metoda rekurzivního sestupu	5
1.5	Transformace gramatik na LL(1)	9
1.6	Syntaktická analýza LL(k) gramatik	11
2	Deterministická syntaktická analýza zdola nahoru	13
2.1	Sestrojení LR(0) analyzátoru	16
2.2	Jednoduché LR(k) gramatiky	17
2.3	Analýza LR(k) gramatik	17

1 Deterministická syntaktická analýza shora dolů

1.1 LL(k) gramatiky

LL(k) gramatiky a jejich analýza (from **L**eft to right **L**eftmost parse).

Definice 1.1

Nechť $G = (N, \Sigma, P, S)$ je CFG, $k \geq 1$ celé číslo.

Definujeme funkci $FIRST_k^G(\alpha) : (N \cup \Sigma)^+ \rightarrow \Sigma^{*k}$ takto:

$$FIRST_k^G(\alpha) = \{w \in \Sigma^* \mid (\alpha \Rightarrow^* w \wedge |w| \leq k) \vee (\alpha \Rightarrow^* w\beta \wedge |w| = k)\}$$

Dále definujeme funkci $FOLLOW_k^G : N \rightarrow \Sigma^{*k}$ takto:

$$FOLLOW_k^G(A) = \{w \in \Sigma^* \mid S \Rightarrow^* uA\alpha, w \in FIRST_k^G(\alpha)\}$$

Označení: Nechť $w = a_1 \dots a_n$, $k \geq 1$ celé číslo. klademe

$$k : w = \begin{cases} a_1 \dots a_k & , k < n \\ w & , k \geq n \end{cases}$$

Definice 1.2

Nechť $G = (N, \Sigma, P, S)$ je CFG, $k \geq 1$ celé číslo. Řekneme, že G je LL(k) gramatika, právě když pro libovolné dvě nejlevější derivace

$$(1) S \Rightarrow^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* wx \quad (2) S \Rightarrow^* wA\alpha \Rightarrow w\gamma\alpha \Rightarrow^* wy$$

podmínka $k : x = k : y$ značí $\beta = \gamma$.

G je LL $\Leftrightarrow \exists k : G$ je LL(k)

L je LL(k) $\Leftrightarrow \exists G \exists k : G$ je LL(k) a $L=L(G)$.

Definice 1.3

Nechť $G = (N, \Sigma, P, S)$ je CFG. Jestliže pro každé $A \in N$ platí, že všechna A-pravidla mají pravé strany začínající různými terminálními symboly, pak G nazveme *jednoduchou LL(1) gramatikou*.

Věta 1.4

Nechť $G = (N, \Sigma, P, S)$ je CFG. Pak G je LL(k), právě když platí: Jsou-li $A \rightarrow \beta$ a $A \rightarrow \gamma$ dvě různá pravidla v P, pak pro všechny nejlevější větné formy $wA\alpha$ platí:

$$FIRST_k(\beta\alpha) \cap FIRST_k(\gamma\alpha) = \emptyset.$$

Důkaz: Z definice LL(k) a FIRST.

Předpokládejme, že průnik je neprázdný, tedy existuje x , které tam patří.

Z definice FIRST:

$$S \Rightarrow^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* wx y$$

$$S \Rightarrow wA\alpha \Rightarrow w\gamma\alpha \Rightarrow wx z$$

$k : xy = k : xz$, protože $x \in FIRST_k$.

Protože $\beta \neq \gamma$, G není LL(k).

Předpokládejme, že G není LL(k). Pak existují dvě různé derivace

$$S \Rightarrow^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* wx$$

$$S \Rightarrow^* wA\alpha \Rightarrow w\gamma\alpha \Rightarrow^* wy$$

takové, že $k : x = k : y$, ale $\beta \neq \gamma$, to jest $A \rightarrow \beta$ a $A \rightarrow \gamma$ jsou různé a $FIRST_k(\beta\alpha)$ a $FIRST(\gamma\alpha)$ obsahují řetězec $k : x$ a tedy nejsou disjunktní. \square

Důsledek 1.5

Důsledky definice LL(k)

- Necht' $G = (N, \Sigma, P, S)$ je CFG bez ε pravidel. G je LL(k), právě když

$$\forall A \in N, \forall p \in P, A \rightarrow \alpha_1, \dots, A \rightarrow \alpha_n$$

platí $FIRST_1(\alpha_i) \cap FIRST_1(\alpha_j) = \emptyset$ pro $1 \leq i \neq j \leq n$.

- Necht' $G = (N, \Sigma, P, S)$ je CFG. G je LL(1) gramatika, právě když $\forall A \in N, \forall p_1, p_2 \in P, A \rightarrow \beta, A \rightarrow \gamma$ platí: $FIRST_1(\beta FOLLOW(A)) \cap FIRST_1(\gamma FOLLOW(A)) = \emptyset$

Definice 1.6

Necht' $G = (N, \Sigma, P, S)$ je CFG. G je *silná LL(k) gramatika* (pro $k \geq 1$ celé), jestliže $\forall A \in N, \forall p_1, p_2 \in P, A \rightarrow \beta, A \rightarrow \gamma$ platí: $FIRST_k(\beta FOLLOW_k(A)) \cap FIRST_k(\gamma FOLLOW_k(A)) = \emptyset$

Poznámka 1.7

Každá silná LL(k) (SLL(k)) gramatika je LL(k) gramatikou.

Příklad 1.1

Uvedeme příklad gramatiky, která je LL(2) a není SLL(2).

$$\begin{aligned} S &\rightarrow aAaa \mid bAba \\ A &\rightarrow b \mid \varepsilon \end{aligned}$$

$$FOLLOW_2(S) = \{\varepsilon\}, FOLLOW_2(A) = \{aa, ba\}$$

$$S \text{ } FIRST_2(aAaa.FOLLOW_2(S)) = \{ab, aa\},$$

$$FIRST_2(bAba.FOLLOW_2(S)) = \{bb\},$$

Množiny jsou disjunktní.

$$A \text{ } FIRST_2(b.FOLLOW_2(A)) = \{ba, bb\},$$

$$FIRST_2(\varepsilon.FOLLOW_2(A)) = \{aa, ba\},$$

Množiny nejsou disjunktní, proto není SLL(2).

Poznámka 1.8

Každá SLL(1) je LL(1) a naopak.

1.2 Syntaktická analýza LL(1) gramatik

Syntaktická analýza LL(1) gramatik jde provádět jednostavovým zásobníkovým automatem. Přechodová funkce M je zadána takto:

$$\text{dom}(M) = (N \cup \Sigma \cup \xi) \times (\Sigma \cup \{\varepsilon\})$$

$$\text{im}(M) = \{ \langle \alpha, i \rangle \mid A \rightarrow \alpha \text{ je } i\text{-té pravidlo v } P \} \cup \{ \text{odstraň, přijmi, chyba} \}$$

Je-li $G = (N, \Sigma, P, S)$, pak

1. Je-li $A \rightarrow \alpha$ i -té pravidlo, klademe $M(A, a) = \langle \alpha, i \rangle$ pro všechna $a \in \text{FIRST}_1(\alpha)$. Je-li též $\varepsilon \in \text{FIRST}_1(\alpha)$, pak $M(A, b) = \langle \alpha, i \rangle$ pro všechna $b \in \text{FOLLOW}_1(A)$.
2. $M(a, a) = \text{odstraň}$.
3. $M(\$, \varepsilon) = \text{přijmi}$.
4. $M(x, a) = \text{chyba}$.

Syntaktická analýza:

(vstup, zásobník, výstup), $(ax, A\alpha, \pi i)$

počáteční konfigurace $(w, S\$, \varepsilon)$

\vdash :

1. $(ax, A\alpha, \pi) \vdash (ax, b\alpha, \pi i)$, jestliže $M(A, a) = \langle \beta, i \rangle$
2. $(ax, a\alpha, \pi) \vdash (x, a, \pi)$
3. $(\varepsilon, \$, \pi) \dots$ koncová konfigurace
4. $(x, \alpha, \pi) \vdash \text{"chyba"}$

Věta 1.9

Každá LL(k) gramatika je jednoznačná.

Důkaz: Nechť gramatika G není jednoznačná. pak existuje $w \in L(G)$ tak, že existují dvě různé derivace věty w :

1. $S \Rightarrow^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* wx = \tilde{w}$
2. $S \Rightarrow^* wA\alpha \Rightarrow w\gamma\alpha \Rightarrow^* wy = \tilde{w}$

kde $A \rightarrow \beta$ a $A \rightarrow \gamma$ jsou dvě různá pravidla, $\text{FIRST}_k(x) = \text{FIRST}_k(y)$, ale $\beta \neq \gamma$, tedy G není LL(K). \square

Věta 1.10

Je-li G levorekurzivní, pak není $LL(k)$ pro žádné k .

Důkaz: Buď $A \in N$, A je levorekurzivní.

$$A \Rightarrow^* A\alpha \begin{cases} \alpha \Rightarrow^* \varepsilon \\ \alpha \not\Rightarrow^* \varepsilon \end{cases}$$

V prvním případě je $A \Rightarrow^+ A$ jednoznačné, tedy není $LL(k)$. Ve druhém necht' $\alpha \Rightarrow^* v$, $v \in \Sigma^+$ a dále $A \Rightarrow^* u$.

1. $S \Rightarrow^* wA\alpha' \Rightarrow^* wA\alpha^k\alpha' \Rightarrow^* wv^k\alpha'$
2. $S \Rightarrow^* vA\alpha' \Rightarrow^* wA\alpha^k\alpha'S \Rightarrow wA\alpha^{k+1}\alpha'S \Rightarrow^* wv^{k+1}\alpha'$

$$k : v^k = k : v^{k+1}, uv^k = k : uv^{k+1} \quad \square$$

Příklad 1.2

Gramatika

$$\begin{aligned} E &\rightarrow E + T | T \\ T &\rightarrow T * F | F \\ F &\rightarrow i | (E) \end{aligned}$$

je levorekurzivní a proto není $LL(1)$. Někdy je možné gramatiku upravit.

1.3 Analýza $SLL(k)$ gramatik**Algoritmus 1.1**

Vytvoření $SLL(k)$ tabulky M pro $SLL(k)$ gramatiku $G = (N, \Sigma, P, S)$.

$M : N \cup \$ \times \Sigma^{*k} \rightarrow \{ \langle \alpha, i \rangle \mid i : A \rightarrow \alpha \in P \} \cup \{ \text{chyba} \}$.

- Je-li $i : A \rightarrow \alpha \in P$ i -té pravidlo, pak $M(A, x) = \langle \alpha, i \rangle$ pro všechna $x \in FIRST_k(\alpha)$ taková, že $|x| = k$.
- Je-li $i : A \rightarrow \alpha \in P$ i -té pravidlo, pak $M(A, x) = \langle \alpha, i \rangle$ pro všechna $x \in FIRST_k(FIRST_k(\alpha).FOLLOW_k(A))$.

Algoritmus 1.2

Syntaktická analýza pro $SLL(k)$ gramatiku. Algoritmus \mathcal{A} .

Vstup:

M - $SLL(k)$ tabulka pro danou $SLL(k)$ gramatiku G .

Výstup:

Je-li $w \in L(G)$, pak levý rozbor věty w (ozn. Π), jinak chybové hlášení.

Metoda:

Označme $w = k : x$, kde x je dosud nepřečtená část vstupního řetězce.

1. Počáteční konfigurace je $(w, S\$, \varepsilon)$
2. $(x, A\alpha, \Pi) \vdash (x, \beta\alpha, \Pi i)$, pro $M(A, w) = \langle \beta, i \rangle$
3. $(ax', a\alpha, \Pi) \vdash (x', \alpha, \Pi)$

4. $(\varepsilon, \$, \Pi)$ je koncová konfigurace.
5. jinak $(x, X\alpha, \Pi) \vdash$ chyba.

Definice 1.11

Řekneme, že algoritmus syntaktické analýzy je *správný*, pro bezkontextovou gramatiku $G = (N, \Sigma, P, S)$, jestliže

- $L(G) = \{w \mid \mathcal{A}(w) \text{ je definováno a } \mathcal{A}(w) \neq \text{chyba}\}$.
- Je-li $\mathcal{A}(w) = \Pi$, pak Π je levý rozbor věty w .

Jestliže \mathcal{A} používá tabulku M a \mathcal{A} je správný pro w , pak i M je správná pro w .

Věta 1.12

Algoritmus vytvoření SLL(k) tabulky vytváří správnou tabulku pro \mathcal{A} .

Důkaz: Jestliže G je SLL(k), pak pro libovolný argument M je definována nejvýše jedna položka tvaru $\langle a, i \rangle$.

$$S \Rightarrow^* w \iff (w, S\$, \varepsilon) \vdash^* (\varepsilon, \$, \Pi)$$

Důkaz indukci:

- a) $(xy, S\$, \varepsilon) \vdash^* (y, \alpha\$, \Pi)$, pak $S \Rightarrow^* x\alpha$.
- b) $S \Rightarrow^* x\alpha$, $k : y \in \text{FIRST}_k(\alpha)$, pak $(xy, S\$, \varepsilon) \vdash^* (y, \alpha\$, \Pi)$.

□

1.4 Metoda rekurzivního sestupu

Při metodě rekurzivního sestupu se využívají známé syntaktické diagramy. Vznikající program sestává ze vzájemně rekurzivních procedur, každému neterminálu odpovídá ‘zhruba’ jedna procedura.

Tedy postupně:

- Z neterminálů sestojíme syntaktické diagramy.
- Ze syntaktických diagramů sestojíme rekurzivní procedury.

Poznámka 1.13

Gramatika je v BNF (**B**ackusova **N**ormální **F**orma), má-li na pravé straně alternativy tvaru $A \rightarrow aC \mid bD \mid \dots$

GBNF (zobecněná BNF) přidává do syntaxe zápisu gramatiky symboly $+$, $\{\dots\}$ a $(\dots)^*$ pro iterace apod.

Algoritmus 1.3

Vytvoření syntaktických diagramů z gramatiky v GBNF

Vstup:

Gramatika v GBNF

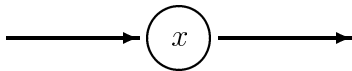
Výstup:

syntaktické diagramy

Metoda:

Pro každé $A \in N$, $A ::= \xi_1 | \dots | \xi_n$ vytvoř graf, jehož struktura je určena pravými stranami $\xi_1 | \dots | \xi_n$ dle následujících pěti pravidel.

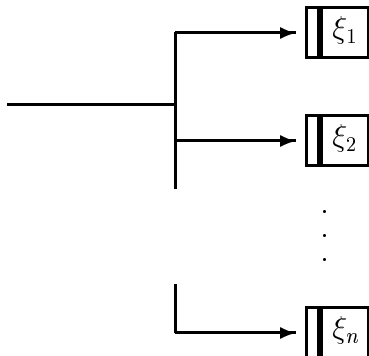
- Každé $x \in \Sigma$ v ξ_i je reprezentováno grafem



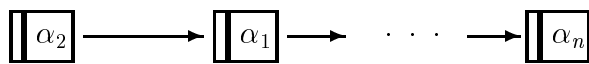
- Každé $B \in N$ v ξ_i je reprezentováno grafem



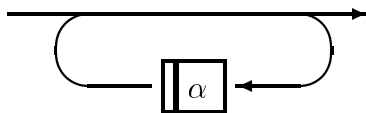
- $A ::= \xi_1 | \dots | \xi_n$ jsou zobrazena na graf



- $\xi = \alpha_1 \dots \alpha_n$ jsou zobrazena na graf



- $\xi = \{\alpha\}$ je zobrazeno na graf

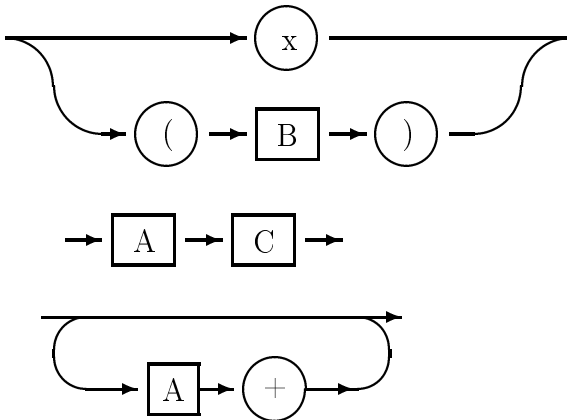


Příklad 1.3

Gramatika v GBNF

$$\begin{aligned} A &::= x|(B) \\ B &::= AC \\ C &::= \{+A\} \end{aligned}$$

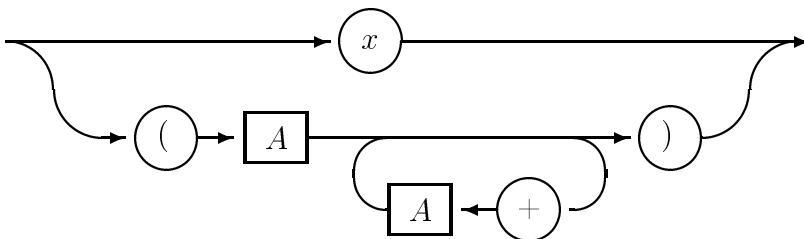
se převede na



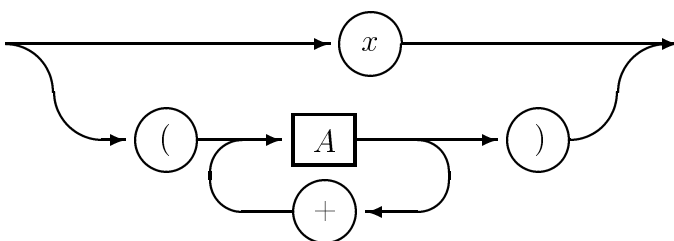
Systém grafů z výstupu algoritmu je vhodné redukovat na dostatečně malý počet rozumně velkých grafů pomocí substituce.

Příklad 1.4

Z předchozího příkladu tak dostaneme graf



A po zjednodušení

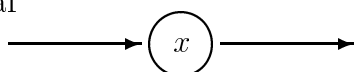


Algoritmus 1.4

Transformace syntaktických diagramů na program

Každý graf převed' pomocí následujících pěti kroků na procedury

- Graf



Převed' na

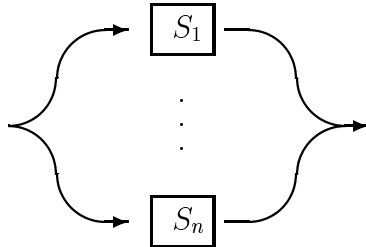
```
if ch='x' then read(ch) else error
```

- Graf



Převéd' na
call A

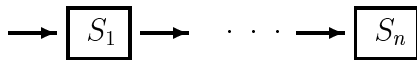
- Graf



Převéd' na
case ch of
 $L_1: T(S_1)$
 \vdots
 $L_n: T(S_n)$
endcase

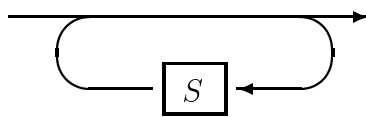
kde $L_i = FIRST(S_i)$, $T(S_i)$ je S_i po transformaci.

- Graf



Převéd' na
begin
 $T(S_1)$;
 \vdots
 $T(S_n)$
end

- Iterace



Se převede na
while ch in L do T(S),
kde $L = FIRST(S)$

Příklad 1.5

program PARSER;

```
var ch:char
  proc A;
  begin
    if ch='x' then
```

```

        read(ch)
    elseif ch='(' then begin
        read(ch);
        A;
        while ch='+' do begin
            read(ch);
            A;
        endwhile;
        if ch=')' then
            read(ch)
        else
            error("očekávám ')")
        endif
    endif
end;
BEGIN
    read(ch);
    A;
END

```

1.5 Transformace gramatik na LL(1)

Transformovat gramatiku na LL(1) znamená odstranit kolize FIRST-FIRST a FIRST-FOLLOW. K odstranění kolize FIRST-FIRST může vést kombinace těchto kroků:

- Odstranění levé rekurze
- Substituce levého rohu

$$\begin{array}{l} A \rightarrow B\alpha \\ B \rightarrow \beta_1 | \dots | \beta_n \end{array} \implies A \rightarrow \beta_1\alpha | \dots | \beta_n\alpha$$

- Levá faktorizace (vytýkání)

$$\begin{array}{l} A \rightarrow \alpha\alpha_1 | \dots | \alpha\alpha_n \end{array} \implies \begin{array}{l} A \rightarrow \alpha A' \\ A' \rightarrow \alpha_1 | \dots | \alpha_n \end{array}$$

Příklad 1.6

Při kolizi FIRST-FIRST: $A \rightarrow \alpha\beta$, $FIRST(\alpha) \cap FIRST(\beta) \neq \emptyset$ provedeme levou faktorizaci, je-li to možné. Nechť například $\Sigma = \{a, i\}$, $N = \{S, S_L\}$

$$\begin{array}{l} S_L \rightarrow SiS_L \\ S \rightarrow a \end{array}$$

kolize: $FIRST(S) \cap FIRST(SiS_L) = \{a\}$.

Tuto gramatiku převedeme na

$$\begin{aligned} S_L &\rightarrow SS' \\ S' &\rightarrow \varepsilon | iS_L \\ S &\rightarrow a \end{aligned}$$

Výsledná Gramatika je LL(1).

Příklad 1.7

Nelze-li aplikovat levou faktorizaci, provedeme substituci levého rohu a po ní levou faktorizaci.

$$\begin{aligned} A &\rightarrow aB|CB \\ C &\rightarrow aC|bB \\ B &\rightarrow cB|D \end{aligned}$$

Tato gramatika není LL(1), protože $FIRST(aB) \cap FIRST(CB) = \{a\}$. Po substituci levého rohu dostaneme

$$\begin{aligned} A &\rightarrow aB|aCB|bBB \\ B &\rightarrow cB|d \\ C &\rightarrow aC|bB \end{aligned}$$

Gramatika po levé faktorizaci již bude LL(1):

$$\begin{aligned} A &\rightarrow aA'|bBB \\ A' &\rightarrow B|CB \\ B &\rightarrow CB|d \\ C &\rightarrow aC|bB \end{aligned}$$

V případě kolize FIRST-FOLLOW ($A \rightarrow \alpha|\beta$, $\alpha \Rightarrow^* \varepsilon$, $FOLLOW(A) \cup FIRST(\beta) \neq \emptyset$) se používají tyto transformace:

- Pohlcení terminálu:

$$G = (N, \Sigma, P, S), B \in N, a \in \Sigma, \alpha, \beta \in (N \cup \varepsilon)^*$$

$$\begin{aligned} A &\rightarrow \alpha Ba\beta \\ B &\rightarrow \alpha_1 | \dots | \alpha_n \end{aligned}$$

transformujeme na gramatiku $G' = (N \cup \{[Ba]\}, \Sigma, P', S)$, $P' = \{A \rightarrow \alpha Ba\beta\} \cup \{A \rightarrow \alpha[Ba]\beta\} \cup \{[Ba] \rightarrow \alpha_1 a | \dots | \alpha_n a\}$

- Extrakce pravého kontextu:

$$G = (N, \Sigma, P, S), A, X, Y \in N, \alpha\beta \in (N \cup \Sigma)^*, a \in FIRST(YB). P:$$

$$\begin{aligned} X &\rightarrow \alpha AYB \\ Y &\rightarrow a\gamma_i \\ Y &\rightarrow \delta_j \end{aligned}$$

$i = 1, \dots, m, j = 1, \dots, k, a \in FIRST(\delta_j)$,

$P' = P - \{X \rightarrow \alpha AY\beta\} \cup \{X \rightarrow \alpha Aa\gamma_1\beta | \dots | \alpha Aa\gamma_m\beta | \alpha A\delta_1\beta | \dots | \alpha A\delta_k\beta\}$,

$L(G) = L(G')$.

Příklad 1.8

Gramatiku, která není LL(1)

$$\begin{aligned} A &\rightarrow BDC \\ B &\rightarrow \varepsilon|aaC \\ C &\rightarrow c|bC \\ D &\rightarrow a \end{aligned}$$

Transformujeme na

$$\begin{aligned} A &\rightarrow BaC \\ B &\rightarrow \varepsilon|aaC \\ C &\rightarrow c|bC \end{aligned}$$

Další transformací je pohlcení terminálu, $[Ba] \in N$:

$$\begin{aligned} A &\rightarrow [Ba]C \\ [Ba] &\rightarrow a|aaCa \\ C &\rightarrow c|bC \end{aligned}$$

Levá faktorizace:

$$\begin{aligned} A &\rightarrow [Ba]C \\ [Ba] &\rightarrow aE \\ E &\rightarrow \varepsilon|aCa \\ C &\rightarrow c|bC \end{aligned}$$

A nakonec substituce:

$$\begin{aligned} A &\rightarrow aEC \\ E &\rightarrow \varepsilon|aCa \\ C &\rightarrow c|bC \end{aligned}$$

Výsledná gramatika je LL(1).

1.6 Syntaktická analýza LL(k) gramatik**Definice 1.14**

Nechť Σ je abeceda, $L_1, L_2 \subseteq \Sigma^*$, $k \geq 1$. Definujeme operátor \oplus_k takto:

$$L_1 \oplus_k L_2 = \{w | \exists x \in L_1, \exists y \in L_2 : w = k : xy\}$$

Poznámka 1.15

$$FIRST_k(\alpha\beta) = FIRST_k(\alpha) \oplus_k FIRST_k(\beta).$$

Definice 1.16

Nechť $G = (N, \Sigma, P, S)$ je bezkontextová gramatika. Pro každé $A \in N$ a $L \subseteq \Sigma^{*k}$ definujeme $T_{A,L}$ — LL(k) tabulka pro (A, L) — jako parciální funkci $T_{A,L} : \Sigma^{*k} \rightarrow \{A \rightarrow \alpha_1 | \dots | \alpha_s\} \times P(\Sigma^{*k})$ takto:

1. $T_{A,L}(u) = \text{'error'}$, jestliže neexistuje $A \rightarrow \alpha \in P$: $FIRST_k(\alpha) \oplus_k L$ obsahující řetěz u .
2. $T_{A,L}(u) = (A \rightarrow \alpha, \langle Y_1, \dots, Y_m \rangle)$, jestliže $A \rightarrow \alpha$ je jediné pravidlo v P takové, že $u \in FIRST_k(\alpha) \oplus_k L$ a je-li $\alpha = x_0 B_1 x_1 \dots B_m x_m$, pak $Y_i = FIRST_k(x_i B_{i+1} x_{i+1} \dots B_m x_m) \oplus_k L$ pro $i = 1, 2, \dots, m$.

3. V ostatních případech není $T_{A,L}(u)$ definováno. (Může nastat jen u gramatik, které nejsou LL(k).)

Algoritmus 1.5

Konstrukce LL(k) tabulek pro LL(k) gramatiku

Vstup:

bezkontextová gramatika $G = (N, \Sigma, P, S)$, G je LL(k).

Výstup:

J — množina LL(k) tabulek.

Metoda:

- Inicializace – konstrukce tabulky T_0 LL(k) pro S a $\{\varepsilon\}$. $J := \{T_0\}$.
- Iterace – pro každou tabulku $LL(k) \in J$ s položkou $T(u) = (A \rightarrow x_0B_1x_1 \dots B_mx_m, < Y_1, \dots, Y_m >)$ **do** $J := J \cup \{T_{B_i}, Y_i\}$ pro všechna $i \in \langle 1, m \rangle$. Opakuj, dokud se do J dá přidat nějaká LL(k) tabulka.

Algoritmus 1.6

Konstrukce rozkladové tabulky pro LL(k) gramatiku $G = (N, \Sigma, P, S)$

Vstup:

LL(k) gramatika $G = (N, \Sigma, P, S)$, množina LL(k) tabulek J .

Výstup:

rozkladová tabulka pro G : $(J \cup \Sigma \cup \{\$\}) \times \Sigma^{*k} \rightarrow ((T_i \cup \Sigma)^* \times \{1 \dots |p|\}) \cup \{\text{odstraň, přijmi, chyba}\}$.

Metoda:

- Je-li $i : A \rightarrow x_0B_1x_1 \dots B_mx_m \in P$ a $T_{A,L} \in J$, pak pro všechna u taková, že $T_{A,L}(u) = (A \rightarrow x_0B_1x_1 \dots B_mx_m, < Y_1, \dots, Y_m >)$ klademe $M(T_{A,L}, u) = (x_0TB_1, Y_1 \dots TB_m, Y_mx_m, i)$.
- $M(a, av) = \text{odstraň}$ pro libovolné $v \in \Sigma^{*(k-1)}$.
- $M(\$, \varepsilon) = \text{přijmi}$.
- $M(x, a) = \text{chyba}$ v ostatních případech.

2 Deterministická syntaktická analýza zdola nahoru

Definice 2.1

Nechť $G = (N, \Sigma, P, S)$ je bezkontextová gramatika s pravidly očíslovanými $1 \dots p$, $p = |P|$ a $S \Rightarrow_R^{i_1} \alpha_1 \Rightarrow_R^{i_2} \dots \Rightarrow_R^{i_n} \alpha_n = w$ je pravá derivace věty w .

Pak posloupnost čísel pravidel i_n, \dots, i_2, i_1 nazveme *pravý rozbor věty w* .

Poznámka 2.2

Přechod od α_{i+1} k α_i má být deterministický.

Možné typy nedeterminismů jsou

- čtení \times redukce

$$\begin{aligned} A &\rightarrow \alpha.\beta \\ B &\rightarrow \alpha. \end{aligned}$$

- redukce \times redukce

$$\begin{aligned} A &\rightarrow \alpha\beta \\ B &\rightarrow \beta \end{aligned}$$

Definice 2.3

Nechť $G = (N, \Sigma, P, S)$ je bezkontextová gramatika. *Přidruženou gramatikou* ke gramatice G nazveme gramatiku $G' = (N \cup \{S'\}, \Sigma, P \cup \{S' \rightarrow S\}, S')$, kde $S' \notin N \cup \Sigma$.

Definice 2.4

Nechť $G = (N, \Sigma, P, S)$ je bezkontextová gramatika, G' její přidružená gramatika, $k \geq 0$ celé. Řekneme, že G je LR(k) gramatikou, právě když následující tři podmínky:

1. $S' \Rightarrow_R^* \alpha Aw \Rightarrow_R \alpha \beta w$
2. $S' \Rightarrow_R^* \gamma Bx \Rightarrow_R \alpha \beta y$
3. $k : w = k : y$

značí, že $\alpha Ay = \gamma Bx$ (tedy $\alpha = \gamma$, $A = B$, $y = x$).

Poznámka 2.5

Definice postihuje oba typy nedeterminismu.

Definice 2.6

Nechť $G = (N, \Sigma, P, S)$ je bezkontextová gramatika. *položkou gramatiky G* nazveme každý výraz tvaru $A \rightarrow \alpha.\beta$ pro $A \rightarrow \alpha\beta \in P$. Jestliže $\beta = \varepsilon$, pak výraz $A \rightarrow \alpha.$ nazveme *úplná položka*.

Nechť dále $\omega \in (N \cup \Sigma)^*$. Řekneme, že $A \rightarrow \alpha.\beta$ je *platná položka* pro řetěz ω , právě když existuje pravá větná forma ηAu ($u \in \Sigma^*$) taková, že $\eta\alpha = \omega$. Řetěz ω nazveme v tomto případě *platná (perspektivní) předpona (viable prefix)*.

Množinu všech platných položek pro řetěz γ budeme značit $I(\gamma)$.

Poznámka 2.7

Směřujeme k tvrzení, že množina platných položek tvoří regulární jazyk.

Lemma 2.8

Jestliže $A \rightarrow \alpha.B\beta \in I(\gamma)$, pak též $B \rightarrow v \in I(\gamma)$ pro všechna pravidla $B \rightarrow v \in P$.

Důkaz: Protože $A \rightarrow \alpha.B\beta \in I(\gamma)$, existuje pravá větná forma δAw taková, že $\delta\alpha = \gamma$. $\delta\alpha B\beta w \Rightarrow_R^* \delta\alpha Bvw$ je pravá větná forma a tedy $B \rightarrow .v$ je platná položka pro řetěz $\delta\alpha = \gamma$, to jest $B \rightarrow .v \in I(\gamma)$. (operace uzávěru) \square

Lemma 2.9

Jestliže platná položka $B \rightarrow .\beta \in I(\gamma x)$, $x \in N \cup \Sigma$, pak existuje $c \rightarrow \alpha x.\delta \in I(\gamma x)$ taková, že $\delta \Rightarrow_R^* Bz$, $z \in \Sigma^*$.

Důkaz: $B \rightarrow .\beta \in I(\gamma x)$, tedy z definice existuje pravá větná forma ηBu , $\eta = \gamma x$. Pak existuje i pravá větná forma ρCv a pravidlo $C \rightarrow \alpha x.\delta \in P$ takové, že $\rho\alpha x = \gamma x$, a tedy $C \rightarrow \alpha x.\delta$ je platná položka pro $\rho\alpha x = \gamma x$ \square

Lemma 2.10

Jestliže $B \rightarrow \alpha.\beta \in I(\gamma x)$ a $\alpha \neq \varepsilon$, pak $\alpha = \alpha'x$ a $B \rightarrow \alpha'.x\beta \in I(\gamma)$.

Důkaz: Z předpokladu $B \rightarrow \alpha.\beta \in I(\gamma x)$ plyne existence pravé větné formy ηBv takové, že $\eta\alpha = \gamma x$. Protože $\alpha \neq \varepsilon$, $\alpha'x = \alpha$, tedy $\eta\alpha' = \gamma$. \square

Důsledek 2.11

Jestliže $I(\gamma x) \neq \emptyset$, pak $I(\gamma) \neq \emptyset$.

Lemma 2.12

(operace následníka)

Jestliže $A \rightarrow \alpha.x\beta \in I(\gamma)$, pak $A \rightarrow \alpha x.\beta \in I(\gamma x)$

Důkaz: Jestliže $A \rightarrow \alpha.\beta$ je platná pro γ , existuje pravá větná forma ηAu , $\eta\alpha = \gamma$, pak $\eta\alpha x = \gamma x$, tedy $A \rightarrow \alpha x.\beta$ je platná pro $\gamma x \in I(\gamma x)$. \square

Lemma 2.13

Pro libovolnou $\gamma \in (N \cup \Sigma)^*$ a $x \in (N \cup \Sigma)$ lze $I(\gamma x)$ určit na základě znalosti $I(\gamma)$ takto:
 $I(\gamma x)$ je nejmenší množina M taková, že

1. $\{Y \rightarrow \alpha x.\beta \mid Y \rightarrow \alpha.x\beta \in I(\gamma)\} \subseteq M$
2. jestliže $A \rightarrow \alpha.B\beta \in M$, pak i $B \rightarrow .v \in M$

Důkaz:

- $M \subseteq I(\gamma x)$ libovolná položka splňující 1.,2., (1. viz. lemma 2.12 , 2. viz. lemma 2.8).
- mějme libovolnou množinu M , splňující 1.,2. Zvolme libovolnou položku z $I(\gamma x)$ $B \rightarrow \alpha.\beta$ a ukažme, že patří do M .

Pro $\alpha = \varepsilon$ viz. lemma 2.9, pro $\alpha \neq \varepsilon$ viz. lemma 2.10.

\square

Věta 2.14

Nechť $G = (N, \Sigma, P, S)$ je libovolná bezkontextová gramatika. Na $(N \cup \Sigma)^*$ definujeme relaci \sim takto: $\gamma_1 \sim \gamma_2$, právě když $I(\sigma_1) = I(\sigma_2)$. Pak \sim je pravou kongruencí konečného indexu (Relací ekvivalence zprava invariantní konečným indexem).

Důkaz:

- Je zřejmé, že jde o ekvivalenci, je definována pomocí rovnosti.
- \sim má konečný index, protože existuje jen konečně mnoho různých množin položek pro gramatiku G .
- $\gamma_1 \sim \gamma_2 \Rightarrow I(\gamma_1 x) = I(\gamma_2 x)$ plyne z předchozího lemmatu.

□

Lemma 2.15

$I(\varepsilon)$ je nejmenší množina M taková, že

1. $\{S \rightarrow .\alpha \mid S \rightarrow \alpha\} \subseteq M$
2. jestliže $A \rightarrow .B\beta \in M$, pak i $B \rightarrow .\vartheta \in M$ pro každé $B \rightarrow \vartheta \in P$.

Důkaz: Viz. lemma 2.13.

□

Definice 2.16

Nechť $G = (N, \Sigma, P, S)$ je libovolná bezkontextová gramatika. Položkovým automatem pro G nazveme pěticí $A = (Q, \Gamma, \delta, q_0, F)$, kde

- $Q = \{I(\gamma) \mid \gamma \in (N \cup \Sigma)^*\}$
- $\Gamma = N \cup \Sigma$
- $q_0 = I(\varepsilon)$
- $\delta : \delta(I(\gamma), x) = I(\gamma x)$
- $F : Q - \{\emptyset\}$

Příklad 2.1

$$\begin{aligned} S &\rightarrow aAb \\ A &\rightarrow AcB \\ A &\rightarrow B \\ B &\rightarrow d \end{aligned}$$

	Položky	Následníci
$S_0 \equiv I(\varepsilon)$	$S \rightarrow .aAB$	$a : S_1$
$S_1 \equiv I(a)$	$S \rightarrow a.Ab$	
	$A \rightarrow .AcB$	$A : S_2$
	$A \rightarrow .B$	$B : S_3$
	$B \rightarrow .d$	$d : S_4$
$S_2 \equiv I(aA)$	$S \rightarrow aA.b$	$b : S_5$
	$A \rightarrow A.cB$	$c : S_6$
$S_3 \equiv I(aB)$	$A \rightarrow B.$	'redukuj'
S_4	$B \rightarrow d.$	'redukuj'
S_5	$S \rightarrow aAb.$	'redukuj'
S_6	$A \rightarrow Ac.B$	$B : S_7$
	$B \rightarrow .d$	$d : S_4$

2.1 Sestrojení LR(0) analyzátoru

Algoritmus 2.1

Výpočet úplné množiny LR(0) stavů

Operace uzávěru.

function UZ(S)

begin

UZ:=S

repeat

forall $A \rightarrow \alpha.B\beta \in S, B \rightarrow \omega \in P, B \rightarrow .\omega \notin UZ$

do $UZ := UZ \cup \{B \rightarrow .\omega\}$

until žádná další položka se do UZ nedá přidat.

return(UZ)

end

Operace následníka.

function Nasl(S,X)

Nasl(S,X):=UZ($\{A \rightarrow \alpha.x.\beta \mid A \rightarrow \alpha.x\beta \in S\}$)

BEGIN

$C := \{UZ(S' \rightarrow .S)\};$

repeat

forall $Z \in C, X \in N \cup \Sigma: Nasl(Z, X) \neq \emptyset \wedge Nasl(Z, X) \notin C$

do $C := C \cup \{Nasl(Z, X)\}$

until žádná další množina položek se do C nedá přidat

return(C)

END

Definice 2.17

Stav položkového automatu je *neadekvátní*, jestliže platí:

- tento stav obsahuje alespoň jednu úplnou položku $A \rightarrow \alpha$. a alespoň jednu neúplnou položku $B \rightarrow \beta.\gamma$, kde $1 : \gamma \in \Sigma$
- Tento stav obsahuje alespoň dvě různé úplné položky.

Poznámka 2.18

Lze ukázat, že gramatika G je LR(0), právě když položkový automat pro G nemá žádné neadekvátní stavy.

2.2 Jednoduché LR(k) gramatiky**Definice 2.19**

Bezkontextovou gramatiku $G = (N, \Sigma, P, S)$ nazveme *jednoduchou LR(k) gramatikou*, jestliže pro množinu L stavů položkového automatu gramatiky G a dvě různé položky $A \rightarrow \alpha.\beta$ a $B \rightarrow \gamma.\delta$ z q , $q \in L$ platí alespoň jedna z následujících podmínek:

1. $\beta \neq \varepsilon$, $\delta \neq \varepsilon$
2. $\beta = \varepsilon$, $\delta \neq \varepsilon$ a $1 : \delta \in \Sigma$, $FOLLOW_k(A) \cap FIRST_k(\delta.FOLLOW_k(B)) = \emptyset$
3. $\beta \neq \varepsilon$, $\delta = \varepsilon$ a $1 : \beta \in \Sigma$, $FOLLOW_k(B) \cap FIRST_k(\beta.FOLLOW_k(A)) = \emptyset$
4. $\beta = \varepsilon$, $\delta = \varepsilon$ a $FOLLOW_k(A) \cap FOLLOW_k(B) = \emptyset$

2.3 Analýza LR(k) gramatik**Definice 2.20**

Nechť $G = (N, \Sigma, P, S)$ je bezkontextová gramatika, $k \geq 0$ celé. Libovolnou dvojici $\langle A \rightarrow \alpha.\beta, L \rangle$, kde $A \rightarrow \alpha.\beta$ je položka gramatiky G v $L \subseteq \Sigma^{*k}$ nazveme *k-položkou gramatiky G*.

Řekneme, že k-položka $\langle A \rightarrow \alpha.\beta, L \rangle$ je *platná* pro řetěz ω , $\omega \in (N \cup \Sigma)^*$, právě když existuje pravá větná forma ηAu taková, že $\eta\alpha = \omega$ a $k : u \in L$. Množinu všech k-položek platných pro řetěz γ označme $I_k(\gamma)$.

Definice 2.21

Nechť $G = (N, \Sigma, P, S)$ je bezkontextová gramatika, $k \geq 0$ celé, G' je přidružená gramatika. Nechť \mathcal{K}' je množina všech položek pro gramatiku G' a definujme přechodovou funkci $\delta' : \mathcal{K}' \times (N \cup \Sigma) \rightarrow \mathcal{K}'$ takto:

pro každé $K \in \mathcal{K}'$, $x \in (N \cup \Sigma)$ je $\delta'(K, x)$ nejmenší množina M taková, že

1. $M \supseteq \{ \langle Y \rightarrow \alpha x.\beta, L \rangle \mid \langle Y \rightarrow \alpha.x\beta, L \rangle \in K \}$
2. jestliže $\langle A \rightarrow \alpha.B\beta, L \rangle \in M$, pak pro každé pravidlo $B \rightarrow \omega \in P$, $\langle B \rightarrow \omega, FIRST_k(\beta.L) \rangle \in M$.

Dále označme $K_0 \in \mathcal{K}'$ jako nejmenší množinu M takovou, že

1. $M \supseteq \{ \langle S' \rightarrow \cdot S, \perp \rangle \}$

2. jestliže $\langle A \rightarrow .B\beta.L \rangle \in M$, pak pro každé pravidlo $B \rightarrow \omega \in P$, $\langle B \rightarrow .\omega, FIRST_k(B.L) \rangle \in M$.

Konečně označme \mathcal{K} množinu všech prvků z \mathcal{K}' dosažitelných z K_0 pomocí předchozí funkce δ' . nechť dále δ značí restrikcí δ' na $\mathcal{K} \times (N \cup \Sigma)$. Konečný automat $\mathcal{A} = (\mathcal{K}, N \cup \Sigma, \delta, K_0, F)$, kde $F = \mathcal{K} - \{\emptyset\}$ nazveme *k-položkovým automatem pro G* (charakteristickým konečným LR(k) automatem).

Poznámka 2.22

Lze ukázat, že G je $LR(k)$, právě když platí

1. Jsou-li $\langle A \rightarrow \alpha., L_1 \rangle, \langle B \rightarrow \beta., L_2 \rangle \in K$, ($K \in \mathcal{K}$) takové, že $L_1 \cap L_2 = \emptyset$, pak $A \rightarrow \alpha = B \rightarrow \beta$.
2. Jsou-li $\langle A \rightarrow \alpha., L_1 \rangle, \langle B \rightarrow \beta_1.\beta_2, L_2 \rangle \in K$, ($K \in \mathcal{K}$), kde $1 : \beta_2 \in \Sigma$, pak $L_1 \cap FIRST_k(\beta_2.L_2) = \emptyset$.

Definice 2.23

Nechť $G = (N, \Sigma, P, S)$ je bezkontextová gramatika, $k \geq 0$ celé. Definujme funkci EFF_k (**E**mpy-**F**ree **F**irst) takto:

$$EFF_k(\omega) = \{t \in \Sigma^{*k} \mid t \in FIRST_k(\omega) \wedge \omega \Rightarrow^* \beta \Rightarrow^* tv, \text{ kde } \beta \neq Atv\}.$$

Algoritmus 2.2

Algoritmus konstrukce k-položkového automatu pro G

Vstup:

Přidružená gramatika G'

Výstup:

k-položkový automat pro G

Metoda:

```

function Uz(in Z)
begin
  repeat
    forall  $\langle A \rightarrow \alpha.\beta, u \rangle, B \rightarrow w \in P, v \in FIRST_k(\beta.u) : \langle B \rightarrow \omega, v \rangle \notin Z$ 
      do  $Z := Z \cup \{\langle B \rightarrow .\omega, v \rangle\}$ 
    until (žádná nová k-položka se do Z nedá přidat)
  return(Z)
end

function Nasl(Z,X)
begin
   $J := \{\langle A \rightarrow \alpha x.\beta, u \rangle \mid \langle A \rightarrow \alpha.x\beta, u \rangle \in Z\}$ 
  return(Uz(J))
end

```

```

begin
   $\mathcal{K} := \{ \{ \langle S' \rightarrow .S, \perp \rangle \} \}$ 
   $\mathcal{K} := \{ \text{Uz}(\mathcal{K}) \}$ 
  repeat
    forall  $Z \in \mathcal{K}, A \in N \cup \Sigma : \text{Nasl}(Z, A) \neq \emptyset$  and  $\text{Nasl}(Z, A) \notin \mathcal{K}$ 
    do  $\mathcal{K} := \mathcal{K} \cup \{ \text{Nasl}(Z, A) \}$ 
  until žádný nový stav se do  $\mathcal{K}$  nedá přidat
end

```

Definice 2.24

Nechť q je stav LR(k) automatu. klademe $\text{jádro}(q) = \{ A \rightarrow \alpha.\text{beta} \mid \langle A \rightarrow \alpha.\text{beta}, L \rangle \in q \}$.

Metoda konstrukce:

1. Zkonstruuji LR(k) automat $\mathcal{K} = \{ K_0, K_1, \dots, K_n \}$.
2. Pro každé jádro v LR(k) automatu najdi všechny stavy s tímto jádrem a nahraď je jejich sjednocením.
3. Nechť $Y' = \{ Y_0, \dots, Y_m \}$ je výsledná množina stavů po kroku 2. Analyzační akce čti/redukuj se převedou přímo z LR(1) automatu. Nevzniknou-li takto analyzační konflikty, pak G je LALR(k), jinak G není LALR(k).
4. Funkci Nasl vytvoříme takto: Nechť stav Y vznikl sjednocením LR(1) stavů K_{i_1}, \dots, K_{i_r} , $X \in N \cup \Sigma$, $\text{Nasl}(Y, X) = K$, kde K je sjednocením všech stavů, které mají stejné jádro jako $\text{Nasl}(K_{i_j}, X)$, $j \in \langle 1, r \rangle$.

Poznámka 2.25

$\{ \text{stavy LALR} \} = \{ \text{stavy LR}(0) \}$.

Definice 2.26

Nechť $G = (N, \Sigma, P, S)$. Označme Q množinu všech stavů LR(0) automatu a Q' množinu všech stavů LR(k) automatu. *Přesný pravý kontext* LR(0) položky $A \rightarrow \alpha.\beta$ ve stavu $q \in Q$ definujeme takto:

$$\text{ERC}_k(q, \langle A \rightarrow \alpha.\beta \rangle) = \{ t \in \Sigma^{*k} \mid \exists q' \in Q' : q = \text{jádro}(q') \wedge \langle A \rightarrow \alpha.\beta, t \rangle \in q' \}$$

ERC znamená **Exact Right Context**.

Definice 2.27

Nechť $G = (N, \Sigma, P, S)$ je bezkontextová gramatika, $k \geq 1$ celé, Q množina stavů LR(0) automatu pro gramatiku G . Řekneme, že G je LALR(k), jestliže $\forall q \in Q, \forall p \in P$ jsou následující množiny vzájemně po dvou disjunktní:

$$S_q = \{ t \mid \langle A \rightarrow \alpha.\beta \rangle \in q, \beta \neq \varepsilon, t \in \text{EFF}_k(\beta.\text{ERC}_k(q, \langle A \rightarrow \alpha.\beta \rangle)) \}$$

$$S_{q,p} = \text{ERC}_k(q, \langle p : A \rightarrow \alpha. \rangle)$$

Poznámka 2.28

G je LALR(1), právě když pro $A \rightarrow \alpha.\beta$ a $B \rightarrow \gamma.\delta$

1. $\beta \neq \varepsilon \wedge \delta \neq \varepsilon$
2. $\beta \neq \varepsilon \wedge \delta = \varepsilon \wedge FIRST(\beta.LA(q, A)) \cap LA(q, B) = \emptyset$
3. $\beta = \varepsilon \wedge \delta \neq \varepsilon \wedge FIRST(\delta.LA(q, B)) \cap LA(q, A) = \emptyset$
4. $\beta \neq \varepsilon \wedge \delta = \varepsilon \wedge LA(q, A) \cap LA(q, B) = \emptyset$

Algoritmus 2.3

Určení pravých kontextů generovaných a pravých kontextů šířených pro $k = 1$.

Vstup:

Báze stavů LR(0) automatu pro G , označíme ji I .

Výstup:

- Právě kontexty generované položkami ve stavu I pro jádro stavu $Nasl(I, X)$.
- Právě kontexty šířené položkami ve stavu I pro jádro stavu $Nasl(I, X)$.

Metoda:

nechť $\# \notin N \cup \Sigma$. Pravý kontext \perp je pro položku $S' \rightarrow .S$ generován.

forall položky $\langle B \rightarrow \gamma.\delta \rangle \in I$ **do**

begin

$J := UZ(\{\langle B \rightarrow \gamma.\delta, \# \rangle\})$

if $\langle A \rightarrow \alpha.X\beta, a \rangle \in J \wedge a \neq \#$ **then**

a je generován pro položku $\langle A \rightarrow \alpha.X.\beta \rangle \in Nasl(I, X)$;

if $\langle A \rightarrow \alpha.X\beta, \# \rangle \in J$ **then**

pravý kontext se dědí (šíří) z položky $\langle B \rightarrow \gamma.\delta \rangle \in I$

na položku $\langle A \rightarrow \alpha.X.\beta \rangle \in Nasl(I, X)$

end

Algoritmus 2.4

Efektivní výpočet přesných pravých kontextů (LA) pro báze položky stavů LALR(1) automatu.

Vstup:

$G = (N, \Sigma, P, S)$

Výstup:

ERC

Metoda:

1. Zkonstruuji bázi LR(0) automatu pro G .

2. Pomocí předchozího algoritmu urči pro každou bázi stavu I a pro každé $X \in N\Sigma$ pravé kontexty, které jsou pro $Nasl(I, X)$ generovány, resp. řízeny.
3. Zříd' tabulku tvaru: řádky budou označeny položkami bází, sloupce budou obsazeny pravými kontexty $ERC(\text{stav}, \text{položka})$.
 - 3.1. inicializuj první sloupec takto: do tabulky zapiš jen generované pravé kontexty, ostatní := \emptyset .
 - 3.2. iterace: (Round-Robin)

Pro každý stav I a každou položku $i \in I$ a symbol X , je-li jejich kontext šířen na položku $j \in Nasl(I, X)$, **do** change := false

for $NewERC := ERC(Nasl(I, X), j) \cup ERC(I, i)$,

if $NewERC \neq ERC(Nasl(I, X), j)$ **then**

$ERC(Nasl(I, X), j) := NewERC$, change := true;

until not change

Literatura

- [1] Chytil, M.: Teorie automatů a formálních jazyků. SPN, Praha 1982
- [2] Češka, M., Beneš, M., Hruška, T.: Překladače. Nakladatelství VUT, Brno 1993